Virtualisation - R4.A.08

Introduction à la virtualisation

Cours 1

Michel Salomon & David Martinet

IUT Nord Franche-Comté
Département d'informatique

basé sur un support de Jean-Patrick Gelas

Plan des savoirs étudiés en R4.A.08

Descriptif détaillé issu du Référentiel du BUT

- Objectifs
  - Comprendre les principes et les enjeux de la virtualisation
  - Être capable de déployer une solution de virtualisation
  - Découvrir les techniques et outils utilisés pour la virtualisation, amenant au déploiement de plateformes facilitant l'intégration et l'administration de services
- Savoirs de référence étudiés
  - Types de virtualisation (serveur, application, réseau...)
  - Outils de virtualisation (hypervision, conteneurs...)
  - Architectures virtualisées

Introduction

Que veut dire virtualiser ?

proposer, par l'intermédiaire d'une couche d'abstraction proche du matériel, une vue multiple d'un matériel unique, en sérialisant les appels concurrents vus de l'extérieur

Analogie avec les processeurs ⇒ virtualiser = augmenter les cœurs

- Cadence de fonctionnement maximale "atteinte" ($\approx$ 6 GHz)
- Naissance du multicœurs
  - Plusieurs cœurs travaillant en parallèle
  - Problème $\rightarrow$ échange des infos entre les coeurs
- *Hyper-threading* (ou *multi-threading*)
  - Utiliser les unités de calcul d'un cœur en permanence
  - Comment
    - Création de 2 processeurs logiques pour un cœur physique
    - Registres de données, de contrôle dupliqués

Core i9 $\rightarrow$ 1 proc., 8 cœurs phys. hyper-threadés $\rightarrow 2 \times 8 = 16$ cœurs log.

Terminologie

- **Système hôte** → *host*
  - l'OS principal de l'ordinateur
- **Système invité** → *guest*
  - l'OS installé au sein d'une machine virtuelle
- **Machine virtuelle** → *Virtual Machine - VM*
  - un ordinateur virtuel qui utilise un système invité
- Ordinateur virtuel aussi appelé
  - **Serveur privé virtuel** → *Virtual Private Server - VPS*
  - ou environnement virtuel → *Virtual Environment - VE*

Intérêts

- Usage optimal des ressources
- Installation, déploiement et migration facile
- Économie sur le matériel
- Sécurisation
- Isolation
- Allocation dynamique
- Diminution des risques

Historique

Idée développée au centre IBM de Cambridge
et de Grenoble en 1972 (VM/CMS - pseudo-machine)

- Mi-90's
  - des émulateurs
  - d'Atari, Amiga, NES, etc.
- Début des années 2000
  - VMware
- Logiciels libre
  - Xen, QEMU, Bochs, etc.
- Logiciels "propriétaires"
  - VirtualPC, VirtualBox, etc.

Domaines d'application de la virtualisation

Différents domaines - D'un haut niveau à un bas niveau

- Virtualisation d'une application
  - le Contexte d'exécution
- Virtualisation des données
  - Fédérer des bases de données distribuées et hétérogènes
  - *Middleware* (logiciel tiers) fournissant un point d'accès unique
- Virtualisation d'un poste de travail (*Desktop*)
  - Hébergement sur un serveur ***V**irtual **D**esktop **I**nfrastructure*
  - Exemples : Citrix, TS2Log, Virtuel Bureau, etc.
- Virtualisation d'un serveur
  - Conteneur, machine virtuelle
- Virtualisation du réseau
  - **V***irtual* **L***ocal* **A***rea* **N***etwork*
- Virtualisation du stockage
  - Fédérer des volumes de stockage en une ressource unique
  - Virtualisation en mode bloc ou en mode fichier

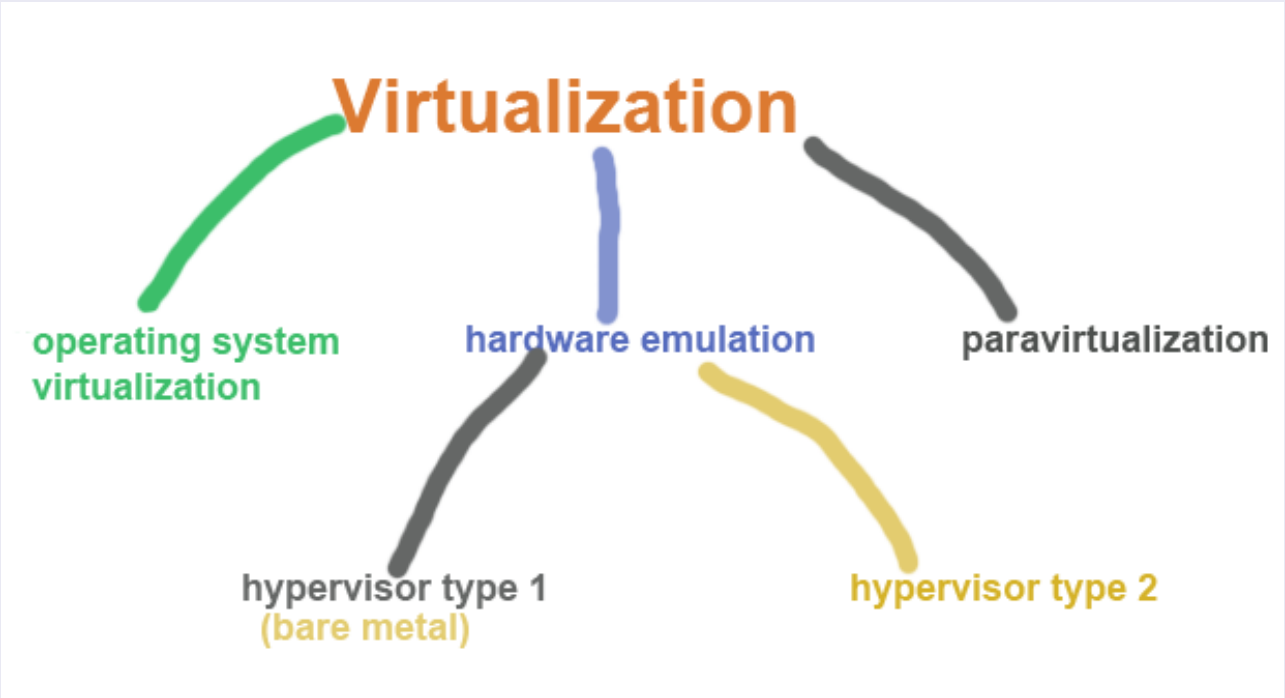
Techniques de virtualisation

Différentes techniques

- Usage de différentes couches logicielles intermédiaires
- 4 types de techno. - Hyperviseur ⇒ plateforme de virtualisation
  1. Virtualisation d'OS ou Isolateur - Conteneur
  2. Hyperviseur de type 2 - Host-metal (architecture hébergée)
  3. Hyperviseur de type 1 - Bare-metal (accès au matériel sans OS)
  4. Paravirtualisation (Hyperviseur de type 1 également)
- Autre techno. plus anecdotique ⇒ noyau dans le *user-space*
  - Noyau exécuté comme une appli. dans l'espace utilisateur
  - Exemple : **U***ser* **M***ode* **L***inux*
    - Avantages
      - Le plantage d'UML n'affecte pas le vrai système hôte
      - Utilisateur pouvant être `root` sans l'être sur le système hôte
      - etc.
    - Inconvénient
      - Très peu performant (empilement de deux noyaux !)
  - Utile au développement noyau

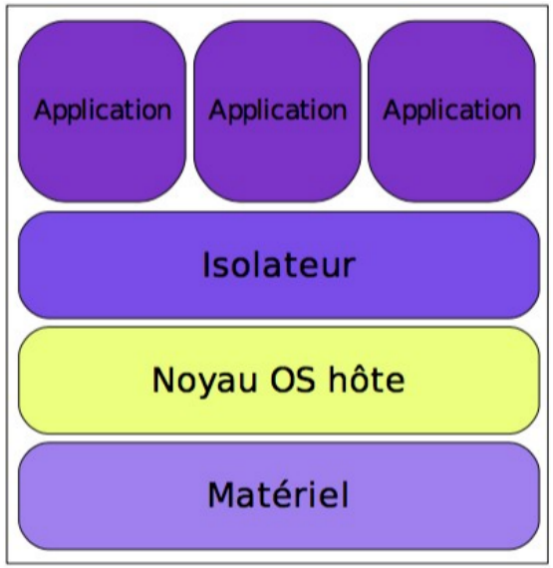
Techniques de virtualisation

Types of virtualization



Techno. 1 - Virtualisation d'OS ou Isolateur - Conteneur

- Isole l'exécution d'applications dans des contextes d'exécution
- Généralisation de la notion de "contexte" Unix, plus isolation
  - des périphériques
  - des systèmes de fichiers
- Performant et économique
- Partage du code du noyau
  $\Rightarrow$ **risque de mauvaise isolation**

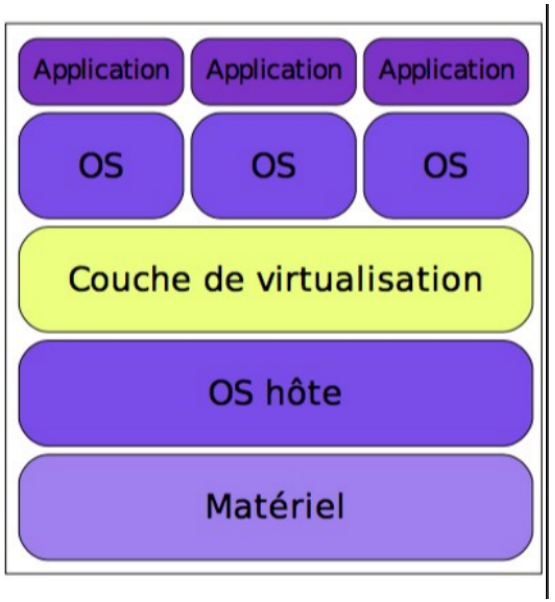


Exemples

- Docker
- LXC - LinXx Container runtime
- OpenVZ - Virtuozzo
- etc.

Techno. 2 - Hyperviseur de type 2

- S'installe sous la forme d'une application dans l'OS hôte
- Virtualise et/ou émule le matériel
- Peut être vu comme un émulateur avec un accès "direct" au proc., la mémoire, le système de fichiers
- Performance réduite si le proc. est émulé
- Bonne étanchéité entre les OS invités

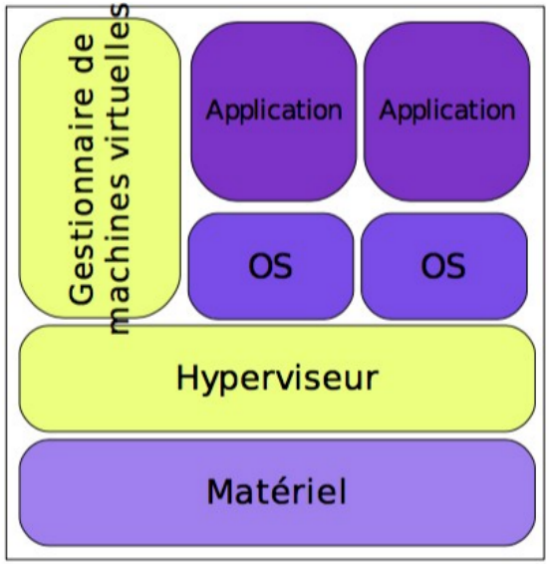


Exemples

- Parallels Desktop
- QEMU
- VirtualBox
- VMware (Desktop hypervisor)
- etc.

Techno. 3 - Hyperviseur de type 1

- Noyau système léger et optimisé
- Outils de supervision
- Permet l'exécution d'OS natifs
- Utilisation d'instructions dédiées à la virtualisation, sinon émulation

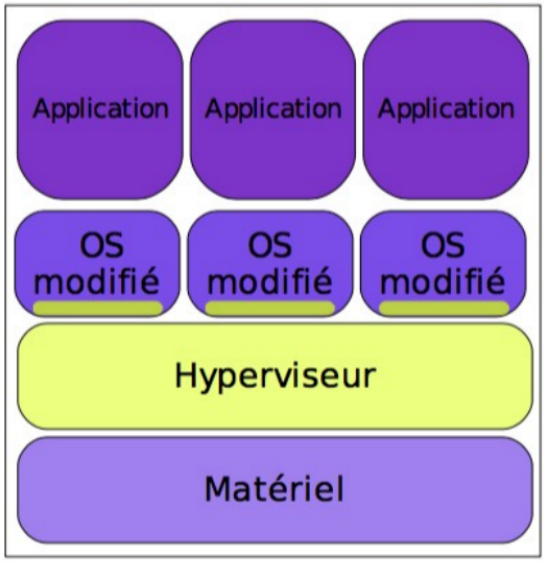


Exemples

- Kernel Virtual Machine
- VMware (vSphere)
- Xen Project
- etc.

Techno. 4 - Paravirtualisation

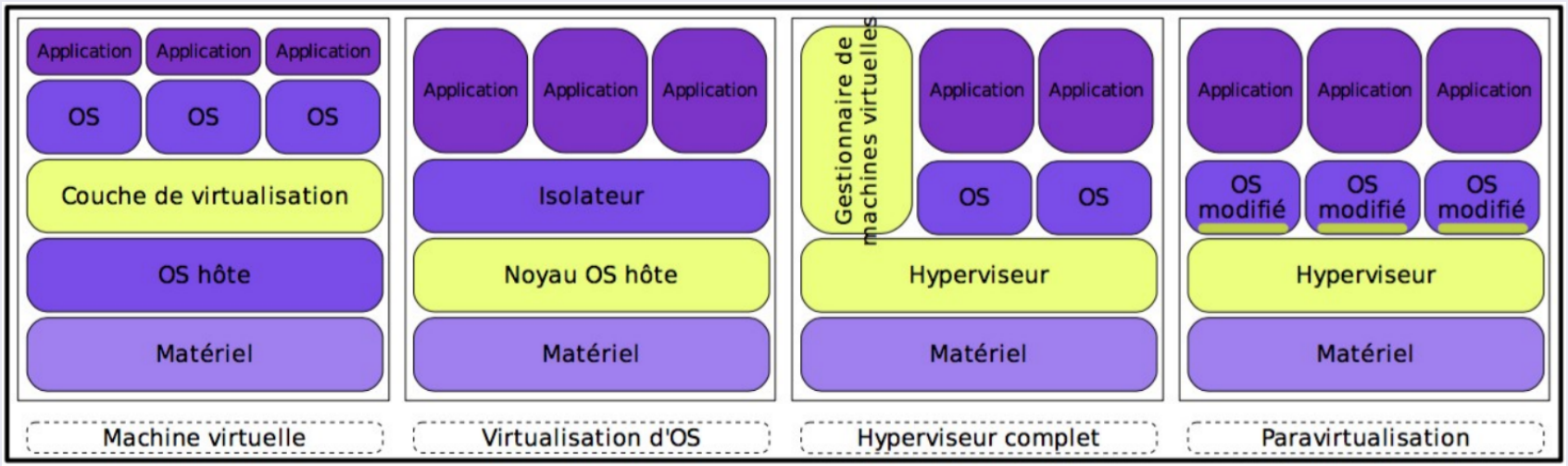
- Noyau système allégé et optimisé
- Noyaux invités adaptés et optimisés
- Utilisation sans instructions spécifiques (par exemple AMD-V ou Intel VT)
- “Impraticable” pour les syst. non libres



Exemples

- Kernel Virtual Machine
- Microsoft Hyper-V
- VMware (vSphere)
- Xen Project
- etc.

Vue comparative des 4 technologies



- *OS-Level Virtualization* $\Rightarrow$ *Container Engine*
- *Full Virtualization*
  - Hyperviseur type 1 ou 2 - *Hardware assisted virtualization (x86 proc.)*
  - Virtualise et/ou émule le matériel
- *Paravirtualization*
  - Hyperviseur type 1 n'émulant pas complètement le matériel
  - Utilise une API pour les appels vers les OS hôtes et virtualisé

Support de la virtualisation - 1

Processeur

- Support pouvant être intégré dans le processeur
  - Protection du processeur physique des accès bas niveaux
  - Virtualisation des accès mémoire
- Simplifie la virtualisation logicielle
- Réduit la dégradation de performance
- Exemples
  - AMD-V
  - Intel VT-x

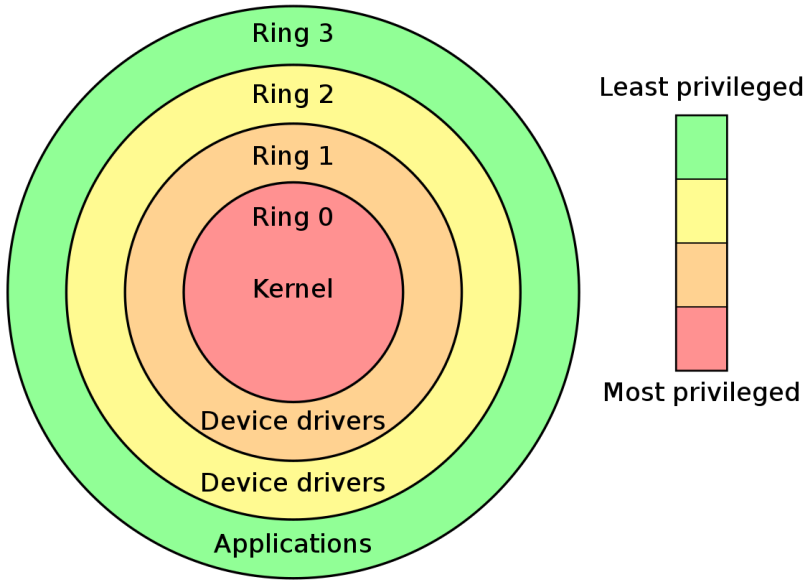
AMD-V et Intel VT

- Jeu étendu d'instructions pour la virtualisation
- Un "super BIOS" fait l'interface avec la puce

Support de la virtualisation - 2-1

x86 Privilege Ring

- 4 niveaux mal exploités
  - *ring 3*
    - Niveau le plus faible
    - Applications
  - *ring 0*
    - Niveau le plus privilégié
    - Système



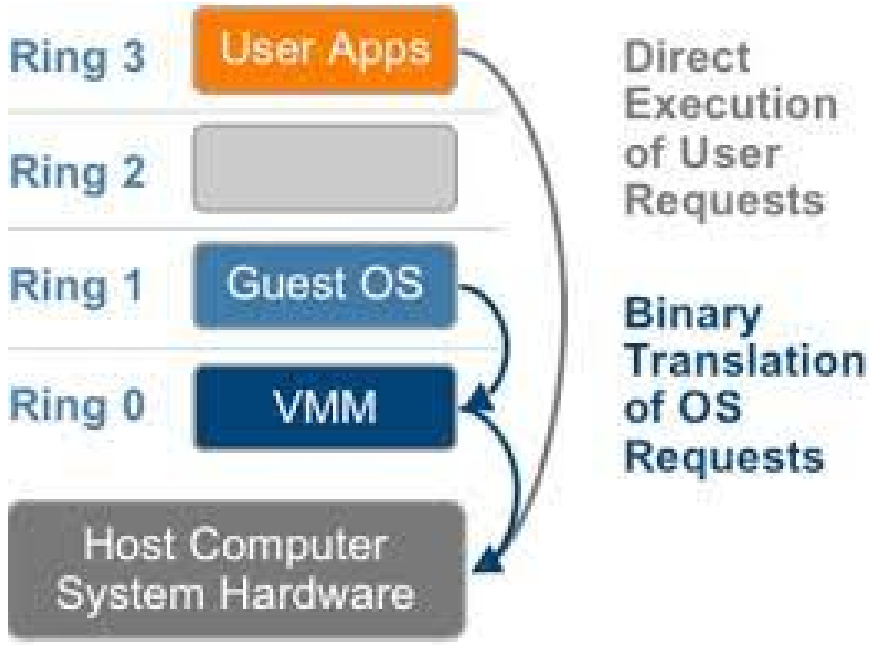
AMD et Intel (vers 2007)

- Ajout d'instructions dédiées pour une virtualisation matérielle
- Coexistence de plusieurs *ring 0* simultanée
- On parle de *ring -1*

Support de la virtualisation - 2-2

Full Virtual. - Binary translation

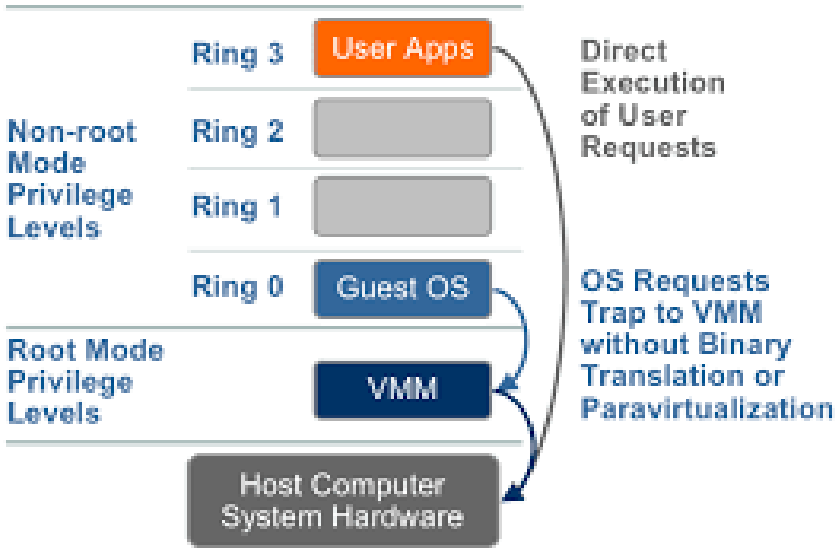
- Utilise de la traduction binaire
- L'hyperviseur intercepte à la volée les appels systèmes pour virtualiser des instructions afin de les remplacer par des instr. produisant l'effet attendu sur le matériel virtuel
- Exécute directement le code utilisateur sur le processeur
- OS invité séparé du matériel par la couche de virtualisation
- OS invité non modifié, ne sait pas qu'il est virtualisé



Support de la virtualisation - 2-3

Full Virtual. - Hardware assisted

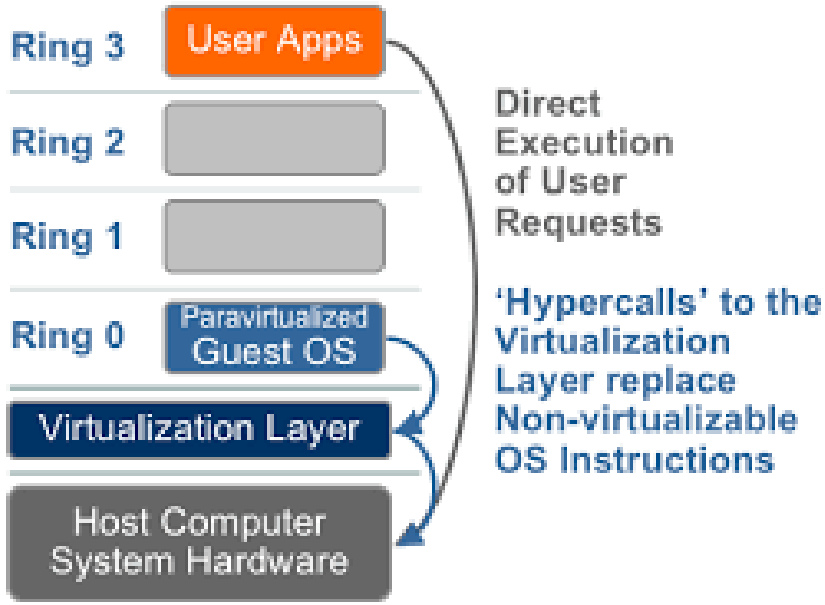
- Intel VT-x et AMD-V permettent des instructions privilégiées avec un nouveau mode d'éxec. pour le processeur
- VMM pouvant s'exécuter dans un niveau plus bas que le ring 0
  - Root mode $\rightarrow$ ring 0P
  - Non-root mode $\rightarrow$ ring 0D
- L'hyperviseur intercepte les appels priviliégés et d'autres pour les traiter directement au niveau matériel



Support de la virtualisation - 2-4

Paravirtualization

- Utilise des “hypercalls”
- L’hyperviseur intercepte les appels de l’OS invité pour les instr. non virtualisables
- L’hyperviseur fournit également des “hypercalls” pour des opér. du noyau (à savoir gestion de la mémoire, des interruptions, du temps)



Support de la virtualisation - 3

Bit NX/XD

- NX signifie *No eXecute* / XD signifie *eXecute Disable* (désignent la même chose)
- Bit spécial permettant de marquer des zones mémoires comme non exécutables
  - Améliore l'isolation des VM
  - Empêche l'exécution de certains types de code
- Fonctionnalité apparue sur les proc. au début des années 2000
  - L'OS (le noyau) doit être adapté pour en tirer avantage
  - Microsoft gère cette techno. depuis Windows XP SP2 et Server 2003 SP1, obligatoire depuis Windows 8
- Voir ce qu'il en est sur un ordinateur exécutant Linux
  - Consulter les *Flags* → `cat /proc/cpuinfo | grep nx | uniq`
  - Consulter les messages → `grep 'Execute' /var/log/messages`

Support de la virtualisation - 4

Flags `LAHF/SAHF`

- *Load AH from Flags* / *Store AH into Flags*
  - Permettent un contrôle direct du registre AH
  - Utilisés par un hyperviseur pour contrôler plus directement le traitement des E/S et les interruptions IRQ
- Fonctionnalité apparue sur les proc. peu après le bit NX/XD
- Liés à l'option `Virtualization Technology (VT)` (soit Intel VT-x ou AMD-V) visible dans le BIOS
- Voir ce qu'il en est sur un ordinateur exécutant Linux
  - Consulter les *Flags* (Drapeaux) $\rightarrow$ `lscpu`

Support de la virtualisation - 5

Virtualisation de la *Memory Management Unit* (MMU)

- *Second Level Address Translation* (SLAT)
  - Intel VT-x → *Extended Page Table* (EPT)
  - AMD-V → *Rapid Virtualization Indexing* (RVI)
- Dans une VM la traduction d'adresses et faite deux fois !
  - Traduction d'adresses obligatoire car les proc. utilisent
    - une Table de pages
    - ou un Tampon de traduction (**T***ranslation* **L***ookaside* **B***uffer*)

    pour convertir les adresses relatives en adresses physiques
- Fonctionnalité apparue avec la génération proc.
  - Nehalem chez Intel
  - Opteron Barcelona chez AMD

Support de la virtualisation - 6

Control groups (Cgroups)

- Fonctionnalité du noyau Linux
  - Interface permettant de limiter, compter et isoler les ressources (proc., mémoire, disque, etc.)
    - Limitation des ressources → ne pas dépasser une limite configurée (mémoire, cache du système de fichiers, etc.)
    - Priorisation → obtenir une plus grande part de l'utilisation du proc. ou des E/S du disque
    - Comptabilité → mesure l'utilisation des ressources par un groupe (pour de la facturation par exemple)
    - Contrôle → geler des processus, les redémarrer
  - allant du contrôle d'un simple processus (avec `nice` par exemple) à celui de la Virtualisation d'OS (Isolateur - Conteneur)
    - Un cgroup est un ensemble de processus liés et associés à un ensemble de paramètres ou de limites
- Isolation par *namespaces* (espaces de nommage)

Compléments sur quelques solutions

VirtualBox - Hyperviseur de type 2

- Machine virtuelle émulant un ordinateur complet
- Support des instructions de virtualisation
- Solution de virtualisation efficace et simple à utiliser
- Gourmand en ressources

Kernel Virtual Machine - Hyperviseur de type 1

- Entièrement intégré au noyau Linux, donc simple à utiliser
- Un fork de QEMU à la base
- Support de la virtualisation dans les proc. indispensable
- Utilisation possible d'un mécanisme de type Paravirtualisation via Virtio (interface de prog. du noyau virtualisant les E/S)

En résumé, la virtualisation c'est...

Intéressant, "indispensable", mais quand même des inconvénients

- Point de défaillance unique
- Besoin de vraies machines puissantes
  - Ne pas virtualiser un serveur déjà très sollicité (*overhead*)
- Dégradation des performances
  - Considérer l'empilement des couches pour les perf. des appli.
  - Virtualisation inadaptée à certains usages (E/S intenses, etc.)
- Simplification / complexification
  - Réduit les coûts, facilite l'administration
  - mais multiplie les serveurs à gérer
- Deux grandes familles de virtualisation
  - Isolateurs - Conteneurs (`Docker` par exemple)
  - Hyperviseurs (`VirtualBox` par exemple)

Concept indispensable étroitement lié à la réussite du Cloud !